



Atty. Dkt. No. 043034-0180

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Takashi TAKENAKA
Title: LOGIC VERIFICATION AND LOGIC CONE EXTRACTION
TECHNIQUE
Appl. No.: 10/612,193
Filing Date: 07/03/2003
Examiner: Unassigned
Art Unit: Unassigned

CLAIM FOR CONVENTION PRIORITY

Commissioner for Patents
PO Box 1450
Alexandria, Virginia 22313-1450

Sir:

The benefit of the filing date of the following prior foreign application filed in the following foreign country is hereby requested, and the right of priority provided in 35 U.S.C. § 119 is hereby claimed.

In support of this claim, filed herewith is a certified copy of said original foreign application:

Japanese Patent Application No. 2002-195514
filed 07/04/2002.

Respectfully submitted,

Date: August 20, 2003

FOLEY & LARDNER
Customer Number: 22428



22428

PATENT TRADEMARK OFFICE

Telephone: (202) 672-5407
Facsimile: (202) 672-5399

By

Phillip J. Anticola Reg. No. 38,819
for /

David A. Blumenthal
Attorney for Applicant
Registration No. 26,257

日本国特許庁
JAPAN PATENT OFFICE

US

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日

Date of Application:

2002年 7月 4日

出願番号

Application Number:

特願2002-195514

[ST.10/C]:

[JP 2002-195514]

出願人

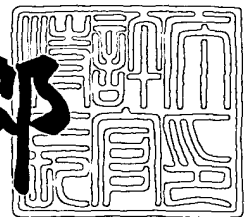
Applicant(s):

日本電気株式会社

2003年 5月 6日

特許庁長官
Commissioner,
Japan Patent Office

太田信一郎



出証番号 出証特2003-3032897

【書類名】 特許願

【整理番号】 34403168

【提出日】 平成14年 7月 4日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 17/50

【発明者】

 【住所又は居所】 東京都港区芝五丁目 7 番 1 号 日本電気株式会社内

 【氏名】 竹中 崇

【特許出願人】

 【識別番号】 000004237

 【氏名又は名称】 日本電気株式会社

【代理人】

 【識別番号】 100088959

 【弁理士】

 【氏名又は名称】 境 廣巳

【手数料の表示】

 【予納台帳番号】 009715

 【納付金額】 21,000円

【提出物件の目録】

 【物件名】 明細書 1

 【物件名】 図面 1

 【物件名】 要約書 1

 【包括委任状番号】 9002136

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 論理検証システム及び方法、論理コーン抽出装置及び方法、論理検証及び論理コーン抽出プログラム

【特許請求の範囲】

【請求項 1】 プログラム言語で記述された動作レベル記述をコンパイルして得た CPU で実行可能なオブジェクトコードから論理コーンを抽出する第 1 の論理コーン抽出手段を備えたことを特徴とする論理検証システム。

【請求項 2】 RT レベル記述から論理コーンを抽出する第 2 の論理コーン抽出手段と、前記第 1 の論理コーン抽出手段で抽出された論理コーンと前記第 2 の論理コーン抽出手段で抽出された論理コーンとの等価性を検証する論理コーン比較手段とを備えたことを特徴とする請求項 1 記載の論理検証システム。

【請求項 3】 前記第 1 の論理コーン抽出手段は、シンボルシミュレーション手段を含むことを特徴とする請求項 1 または 2 記載の論理検証システム。

【請求項 4】 プログラム言語で記述された動作レベル記述をコンパイルして得たオブジェクトコードと、前記動作レベル記述から生成された RT レベル記述と、前記動作レベル記述および前記 RT レベル記述における等価性の比較の対象とする記述の組の情報とこの組毎の等価性の比較の対象となる信号の組の情報とを指定する対応情報と、前記動作レベル記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報とを記憶する記憶手段と、

前記対応情報で指定される前記動作レベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる記述および信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行い、シンボルシミュレーションで得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして抽出する第 1 の論理コーン抽出手段と、

前記対応情報で指定される前記 RT レベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる信号毎の論理コーンを抽出する第 2 の論理コーン抽出手段と、

前記対応情報で指定される前記動作レベル記述と前記 R T レベル記述の比較の対象となる記述毎に、前記第 1 の論理コーン抽出手段で抽出された論理コーンと前記第 2 の論理コーン抽出手段で抽出された論理コーンとを比較する論理コーン比較手段とを備えたことを特徴とする論理検証システム。

【請求項 5】 プログラム言語で記述された動作レベル記述をコンパイルして得た CPU で実行可能なオブジェクトコードから論理コーンを抽出する第 1 の論理コーン抽出手段と、前記動作レベル記述が満たすべき性質であるプロパティを記憶する手段と、前記第 1 の論理コーン抽出手段で抽出された論理コーンをもとに、前記オブジェクトコードが前記プロパティを満たしているかどうかを検査するモデル検査手段とを備えたことを特徴とする論理検証システム。

【請求項 6】 プログラム記述をコンパイルして得たオブジェクトコードと、前記プログラム記述中の論理コーン抽出範囲および該論理コーン抽出範囲毎の抽出対象信号を指定する対応情報と、前記プログラム記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報とを入力する入力手段と、

前記対応情報で指定される論理コーン抽出範囲毎に、前記対応情報で指定される論理コーン抽出範囲および抽出対象信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行うシンボルシミュレーション手段と、

前記シンボルシミュレーション手段で得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして出力する出力手段とを含むことを特徴とする論理コーン抽出装置。

【請求項 7】 プログラム言語で記述された動作レベル記述をコンパイルして得た CPU で実行可能なオブジェクトコードから論理コーンを抽出する第 1 の論理コーン抽出ステップを含むことを特徴とする論理検証方法。

【請求項 8】 R T レベル記述から論理コーンを抽出する第 2 の論理コーン抽出ステップと、前記第 1 の論理コーン抽出ステップで抽出された論理コーンと前記第 2 の論理コーン抽出ステップで抽出された論理コーンとの等価性を検証する論理コーン比較ステップとを含むことを特徴とする請求項 7 記載の論理検証方

法。

【請求項 9】 前記第 1 の論理コーン抽出ステップは、シンボルシミュレーションを含むことを特徴とする請求項 7 または 8 記載の論理検証方法。

【請求項 1 0】 プログラム言語で記述された動作レベル記述をコンパイルして得たオブジェクトコードと、前記動作レベル記述から生成された R T レベル記述と、前記動作レベル記述および前記 R T レベル記述における等価性の比較の対象とする記述の組の情報とこの組毎の等価性の比較の対象となる信号の組の情報とを指定する対応情報と、前記動作レベル記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報とを入力する入力ステップと、

前記対応情報で指定される前記動作レベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる記述および信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行い、シンボルシミュレーションで得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして抽出する第 1 の論理コーン抽出ステップと、

前記対応情報で指定される前記 R T レベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる信号毎の論理コーンを抽出する第 2 の論理コーン抽出ステップと、

前記対応情報で指定される前記動作レベル記述と前記 R T レベル記述の比較の対象となる記述毎に、前記第 1 の論理コーン抽出ステップで抽出された論理コーンと前記第 2 の論理コーン抽出ステップで抽出された論理コーンとを比較する論理コーン比較ステップとを含むことを特徴とする論理検証方法。

【請求項 1 1】 プログラム言語で記述された動作レベル記述をコンパイルして得た C P U で実行可能なオブジェクトコードから論理コーンを抽出する第 1 の論理コーン抽出ステップと、前記動作レベル記述が満たすべき性質であるプロパティを入力するステップと、前記第 1 の論理コーン抽出ステップで抽出された論理コーンをもとに、前記オブジェクトコードが前記プロパティを満たしているかどうかを検査する検査ステップとを含むことを特徴とする論理検証方法。

【請求項 1 2】 プログラム記述をコンパイルして得たオブジェクトコードと、前記プログラム記述中の論理コーン抽出範囲および該論理コーン抽出範囲毎の抽出対象信号を指定する対応情報と、前記プログラム記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報とを入力する入力ステップと、

前記対応情報で指定される論理コーン抽出範囲毎に、前記対応情報で指定される論理コーン抽出範囲および抽出対象信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行うシンボルシミュレーションのステップと、

前記シンボルシミュレーションのステップで得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして出力する出力ステップとを含むことを特徴とする論理コーン抽出方法。

【請求項 1 3】 コンピュータに、プログラム言語で記述された動作レベル記述をコンパイルして得た CPU で実行可能なオブジェクトコードから論理コーンを抽出する第 1 の論理コーン抽出ステップを実行させる論理検証プログラム。

【請求項 1 4】 前記コンピュータに、更に、RTレベル記述から論理コーンを抽出する第 2 の論理コーン抽出ステップと、前記第 1 の論理コーン抽出ステップで抽出された論理コーンと前記第 2 の論理コーン抽出ステップで抽出された論理コーンとの等価性を検証する論理コーン比較ステップとを実行させる請求項 1 3 記載の論理検証プログラム。

【請求項 1 5】 前記第 1 の論理コーン抽出ステップは、シンボルシミュレーションを含むことを特徴とする請求項 1 3 または 1 4 記載の論理検証プログラム。

【請求項 1 6】 コンピュータに、

プログラム言語で記述された動作レベル記述をコンパイルして得たオブジェクトコードと、前記動作レベル記述から生成された RTレベル記述と、前記動作レベル記述および前記 RTレベル記述における等価性の比較の対象とする記述の組の情報とこの組毎の等価性の比較の対象となる信号の組の情報とを指定する対応

情報と、前記動作レベル記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報とを入力する入力ステップと、

前記対応情報で指定される前記動作レベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる記述および信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行い、シンボルシミュレーションで得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして抽出する第 1 の論理コーン抽出ステップと、

前記対応情報で指定される前記 R T レベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる信号毎の論理コーンを抽出する第 2 の論理コーン抽出ステップと、

前記対応情報で指定される前記動作レベル記述と前記 R T レベル記述の比較の対象となる記述毎に、前記第 1 の論理コーン抽出ステップで抽出された論理コーンと前記第 2 の論理コーン抽出ステップで抽出された論理コーンとを比較する論理コーン比較ステップとを実行させる論理検証プログラム。

【請求項 1 7】 コンピュータに、

プログラム言語で記述された動作レベル記述をコンパイルして得た C P U で実行可能なオブジェクトコードから論理コーンを抽出する第 1 の論理コーン抽出ステップと、

前記動作レベル記述が満たすべき性質であるプロパティを入力するステップと、

前記第 1 の論理コーン抽出ステップで抽出された論理コーンをもとに、前記オブジェクトコードが前記プロパティを満たしているかどうかを検査する検査ステップとを実行させる論理検証プログラム。

【請求項 1 8】 コンピュータに、

プログラム記述をコンパイルして得たオブジェクトコードと、前記プログラム記述中の論理コーン抽出範囲および該論理コーン抽出範囲毎の抽出対象信号を指定する対応情報と、前記プログラム記述と前記オブジェクトコードとのマッピン

グ情報を含むコンパイル情報とを入力する入力ステップと、

前記対応情報で指定される論理コーン抽出範囲毎に、前記対応情報で指定される論理コーン抽出範囲および抽出対象信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行うシンボルシミュレーションのステップと、

前記シンボルシミュレーションのステップで得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして出力する出力ステップとを実行させる論理コーン抽出プログラム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は論理回路の設計段階における論理検証に関し、特にCPUで実行可能なプログラム言語で記述された論理回路を対象とした論理検証技術に関する。

【0002】

【従来の技術】

一般に、LSIもしくはVLSIの設計にあたっては、設計を自動的に行う、もしくは設計を支援するLSI設計自動化・支援技術が採用されている。このLSI設計自動化・支援技術を採用した設計のうち、代表的なものとしては、各種EDA(electronic design automation) ツールを利用したトップダウン設計があげられる。この種のトップダウン設計は、その上流工程から、システム設計、機能設計、論理設計、レイアウト設計と大別することができる。このトップダウン設計について図19を参照しながら以下説明する。

【0003】

まず、システムの仕様作成として、LSI全体をシステムと捉え、その動作を記述することが行われる。このようにして作成された記述は、動作レベル記述12と呼ばれる。この動作レベル記述12の作成は、たとえば、自然言語、もしくはVHDL、Verilog-HDLといったハードウェア記述言語HDL(hardw

are description language)、もしくは、C言語、C++言語、j a v a言語などのプログラム言語を用いてなされる。また、S y s t e m C、S p e c Cなどといった、C言語またはC++言語に回路を表現するために便利な特徴を付加したプログラム言語を持ってなされることもある。

【0004】

次に、動作合成フェーズにおいて、作成された動作レベル記述12がRTレベル(register transfer level)記述14に変換される。RTレベル記述14は、通常HDLもしくはプログラム言語を用いてなされる。なお、従来、このフェーズに関しては、人手により置き換えられ、HDLを用いて、直接、RTレベル記述14を作成する場合が多かった。

【0005】

次いで、論理合成フェーズにおいて、RTレベル記述14が、ゲートレベル記述(ゲートレベル論理回路：ネットリスト)1Dに自動変換される。このようにして生成されたネットリストをもとに、レイアウト設計が行われ、ついで、チップ設計が行われる。

【0006】

L S I 設計では、上記のステップによってなされた設計の正当性を調べる設計検証も重要である。設計検証には、たとえば、各合成フェーズの変換が正しいかを検証する等価性検証と、各記述が設計対象を正しく表現しているかを検証する仕様検証とがある。

【0007】

等価検証には、従来シミュレーションによる等価検証もしくは形式的等価性検証が用いられる。

【0008】

シミュレーションによる等価検証は、比較対象とする2つの記述に対して、同一のテストパターンを与えてシミュレーションを行い、その結果が所望の結果と一致するかを確認するものである。シミュレーションは、RTレベル記述14のように記述がHDLで表現されているときは専用のシミュレータを用いて行われる。これに対して、動作レベル記述12のように記述がプログラム言語で表現され

ているときは、たとえばコンパイラを用いてCPUで実行可能な形式（オブジェクトコードとよばれる）に変換された後、得られたオブジェクトコード15をCPUで直接実行して行われる。

【0009】

形式的等価検証は、2つの組み合わせ回路の論理が等価であるか否かを数学的手法により検証するものである。一般的な、RTレベル記述14とゲートレベル記述1Dの等価性を検査する形式的等価性検証装置の構成を示すブロック図が図20に例示される。

【0010】

図20を参照すると、14はRTレベル記述、1Dがゲートレベル記述である。13は両レベルにおいて比較対象となる信号の対応関係を記述した対応情報である。比較対象となる信号は、プライマリ入力、プライマリ出力、レジスタ、その他設計者によって指定される内部端子である。23はRTレベル記述、ゲートレベル記述から、それぞれ論理コーンを抽出する論理コーン抽出手段である。論理コーンとは、信号の論理を決定するために必要な組み合わせ回路のことであり、ブール式で表現される。論理コーンの抽出は、たとえば、文献1：Malik, S. et. al, "Logic Verification Using Binary Decision Diagrams in a Logic Synthesis Environment", Proceedings of IEEE International Conference on Computer-Aided Design, pp.6-9, 1988, に示される方法を用いる。具体的には、注目する変数から入力方向に組み合わせ回路をたどることにより得られる。18、1Eは、それぞれ抽出された論理コーンである。24は、対応する比較対象信号の論理コーンを表現するブール式の論理的な同一性を調べる論理コーン比較手段である。RTレベル記述14とゲートレベル記述1Dで、対応するすべての比較対象信号の論理コーンが論理的に同一であるとき、2つの記述は論理的に等価であると判定し、出力装置3に出力する。

【0011】

図20と同様にRTレベル記述14とゲートレベル記述1Dの等価性を検査する形式的等価性検証装置については特開平8-22485号公報にも記載されている。

【 0 0 1 2 】

仕様検証にも、等価検証と同様に、シミュレーションもしくは形式的手法によるプロパティ検証という技術が知られている。シミュレーションによる仕様検証は、比較対象とする記述に対して、テストパターンを与えてシミュレーションを行い、その結果を解析することにより、回路が所望の機能を有しているかを調べる。形式的手法によるプロパティ検証は、与えられた記述が望ましい性質を満足するかを、静的かつ数学的に検査する方法である。形式的プロパティ検証は、文献 2 : Edmund M. Clarke, et. al. "Model Checking", pp61-74, The MIT Press, 1999に参照される。一般的な、RTレベル記述 1 4 が与えられた性質を満たしていることを検証する形式的プロパティ検証装置の構成を示すブロック図が図 2 1 に例示される。

【 0 0 1 3 】

図 2 1 を参照すると、1 4 は RT レベル記述、1 C は回路が満たすべき性質であるプロパティである。プロパティには、たとえば、「回路がデッドロックを起こさない」、「リクエスト信号が入力されたら一定サイクル以内に応答を返す」といった性質が記述される。論理コーン抽出手段 2 3 は、RT レベル記述 1 4 から論理コーン 1 8 を抽出する。モデル検査手段 2 5 は、論理コーン 1 8 をもとに、回路がとりうる信号の値の組み合わせ（状態と呼ぶ）の集合を順に列挙する。回路がとりうるすべての状態の集合を列挙したところで、これらの集合が与えられたプロパティを満たしているかのチェックを行う。

【 0 0 1 4 】

形式的検証によるプロパティ検証は、シミュレーションに必要なテストデータを必要とせず、また、与えられたプロパティを満たすか否かを完全に検証することが可能であるという特徴を有する。

【 0 0 1 5 】

【発明が解決しようとする課題】

近年の L S I の高集積化・大規模化に伴い、L S I を構成するゲート数は、数百万になることも珍しくなくなっており、設計した回路が意図した機能を有するかを検証する仕様検証をいかに高速に行うかということが重要になってきてい

る。

【 0 0 1 6 】

仕様検証において、従来より一般的に行われているのはシミュレーションによる仕様検証であるが、回路が大規模化するにしたがって、R Tレベル記述 1 4 上でのシミュレーションに多くの時間がかかるようになってきている。これに対して、動作レベルにおける仕様検証は抽象度が高く高速であるという傾向がある。また、プログラム言語で用意された豊富なライブラリを用いてシミュレーションを行うことができる。そのため、特に、図 1 9 に示したように、動作レベル記述 1 2 をプログラム言語記述で表現し、コンパイラを用いて C P U で実行可能なオブジェクトコード 1 5 に変換した後、オブジェクトコード 1 5 を C P U で直接実行して、高速なシミュレーションを実行することが多く行われている。さらに、動作合成フェーズによって、このプログラム言語で表現された動作レベル記述 1 2 から R Tレベル記述 1 4 に自動で変換するということが行われている。

【 0 0 1 7 】

この合成を行うためのツールの実態は、一般に、ソフトウェアで構成される。しかしながら、それらはソフトウェアである以上必ずしもバグがないとはいえず、各合成段階においては、適切な合成がなされたか否かの検証をする必要がある。

【 0 0 1 8 】

従来、最も一般的に行われていた動作合成フェーズにおける正当性の検証は、所定のテストパターンを用いて行われるシミュレーションであったが、このシミュレーションは以下のような問題を含んでいた。

【 0 0 1 9 】

すなわち、テストパターンを用いて行われるシミュレーションを実行するためには、当然ながら、テストパターンの生成を行わなければならない、そのために時間およびコストがかかるといった問題があった。しかも、そのシミュレーションを実行することにより検証できる合成の内容は、テストパターンに依存しており、使用したテストパターンにて考慮されていないエラーを検証することはできない。従って、従来のシミュレーションによる等価検証では、満足する結果を得られない恐れもあった。さらに、動作レベルと R Tレベルの二つのレベルでシミュレ

ーションを繰り返すことは、検証の高速化を目指して動作合成を導入した目的に反するものである。

【 0 0 2 0 】

また、動作合成において動作レベル記述 1 2 から生成された R T レベル記述 1 4 の正当性を検証する他の手法としては、たとえば動作レベル記述 1 2 との形式的等価性検証によって R T レベル記述 1 4 の正当性を検証するものも研究されている。

【 0 0 2 1 】

たとえば、動作合成の検証装置の一例が、文献 3 : 「2000年、フォーマルメソッド・イン・システムデザイン、第16号、59-91頁(Formal Methods in System Design, No. 16, pp.59-91, 2000) 」 に記載されている。

【 0 0 2 2 】

この文献 3 に記載された動作合成の検証装置は、動作レベル記述および R T レベル記述、そして動作合成の副産物である対応関係から、PVSという定理証明システムのための、1)形式仕様記述、2)等価性条件(correctness lemmas), 3)検証スクリプトを出力する。等価性条件とは、動作レベル記述および R T レベル記述の分岐・合流構造を含まない基本ブロック(basic block)毎に実行パス(状態の系列)を抽出し、実行パスの対応関係と信号の対応関係が与えられたときに、各対応するパス毎に対応する信号値の変化が同じであることを表す論理式のことである。PVSは検証スクリプトにしたがって、各パスを実行したときの各信号値の変化を表す関数を項書換えを用いて生成する。そして、対応するパスにおける対応する信号ペアに対して、その変化を表す関数が等価であることを調べる。全ての対応するパスのペア、全ての対応する信号値のペアに対して、その変化を表す関数が等価であることをしめすことにより、動作レベル記述と R T レベル記述が等価であることを検査する。

【 0 0 2 3 】

また、動作合成の検証装置の別の一例が、文献 4 : 「1999年、インターナショナルカンファレンスオンコンピュータデザイン、458-466頁(International Conference on Computer Design, pp. 458-466, 1999) 」 に記載されている。

【 0 0 2 4 】

この文献 4 に記載された動作合成の検証装置は、実行パスを抽出し、実行パスの対応関係と信号の対応関係が与えられたときに、各パスを実行したときの各信号値の変化を表す関数を生成して、対応するパスの対応する信号に対してこの関数が等価であることを調べるのは前出の文献 3 と同じであるが、抽出する実行パスが基本ブロックに制限されない。もとの動作レベル記述の繰り返し構造（ループ構造）に対して、R T レベル記述に対応する実行パスが見つかるまでループを展開しながら実行パスを抽出する。また、信号値の変化を表す関数の生成は、項書換えではなく、シンボルシミュレーションという技術を用いる。

【 0 0 2 5 】

従来は、上述した手法を用いて、専ら、動作レベル記述 1 2 と R T レベル記述 1 4 の等価性を検証しており、動作レベル記述 1 2 をコンパイルして得たオブジェクトコード 1 5 と R T レベル記述 1 4 の等価性の検証については重要視されていなかった。

【 0 0 2 6 】

このため、動作レベル記述 1 2 をオブジェクトコード 1 5 に変換して C P U 上で実行したときの実行結果と、動作レベル記述 1 2 から実際に設計して得られた回路が動作して得られる実行結果とが完全に一致することを保証することができない恐れがあった。すなわち、プログラム言語の多くはコンパイル時にリンクされるライブラリにより言語の意味づけを変更可能であるため、動作レベル記述 1 2 とそれからコンパイルされたオブジェクトコード 1 5 とが等価である保証は必ずしもない。また、動作レベル記述 1 2 を C P U で実行可能な形式に変換するコンパイラもソフトウェアで構成される以上、必ずしもバグがないとはいえず、それが原因で動作レベル記述 1 2 とオブジェクトコード 1 5 とが等価にならない場合もある。若し、動作レベル記述 1 2 とオブジェクトコード 1 5 とが等価でないとすると、動作レベル記述 1 2 から導出された R T レベル記述 1 4 もオブジェクトコード 1 5 と等価でないことになり、オブジェクトコード 1 5 の実行によりその動作の正しさが確認されても、R T レベル記述 1 4 の動作も正しいものと判断することはできない。このため、動作合成フェーズにおいて動作確認が正しく行わ

れていない状況で次の論理合成フェーズに進んでしまう過ちを犯してしまうことになる。

【 0 0 2 7 】

そこで、動作レベル記述 1 2 をコンパイルして得たオブジェクトコード 1 5 を CPU で実行したときの実行結果と、動作合成の結果得られる回路が動作したときの実行結果が完全に一致することを保証できる論理検証システムが望まれる。

【 0 0 2 8 】

また、動作レベル記述 1 2 において形式的プロパティ検証を行いたいという要求もある。この場合、形式的プロパティ検証はシミュレーションと補完する関係にあるため、CPU で実行される動作に対する検証、つまりオブジェクトコード 1 5 に対する検証を行えることが望ましい。

【 0 0 2 9 】

また、これらの論理検証を論理コーンを用いて実現するためには、HDL 記述やゲートレベル記述から論理コーンを抽出する従来の論理コーン抽出技術が使えないため、CPU で実行可能な形式であるオブジェクトコードから論理コーンを抽出する技術が必要である。

【 0 0 3 0 】

【発明の目的】

本発明の目的は、動作レベル記述をコンパイルして得たオブジェクトコードと前記動作レベル記述から導出した RT レベル記述との等価性を検証し得るようにすることで、動作レベル記述を CPU で実行したときの実行結果と、動作合成の結果得られる回路が動作したときの実行結果が完全に一致することを保証できる論理検証システム及び方法を提供することにある。

【 0 0 3 1 】

また本発明の別の目的は、動作レベル記述を CPU で実行したときの動作に対する、形式的プロパティ検証システムおよび方法を提供することにある。

【 0 0 3 2 】

また本発明の他の目的は、オブジェクトコードから論理コーンを抽出する論理コーン抽出装置および方法を提供することにある。

【 0 0 3 3 】

【課題を解決するための手段】

本発明の論理検証システムは、プログラム言語で記述された動作レベル記述をコンパイルして得たCPUで実行可能なオブジェクトコードから論理コーンを抽出する第1の論理コーン抽出手段を備えることを基本とする。そして、RTレベル記述との論理検証に対して本発明を適用する場合、更に、RTレベル記述から論理コーンを抽出する第2の論理コーン抽出手段と、前記第1の論理コーン抽出手段で抽出された論理コーンと前記第2の論理コーン抽出手段で抽出された論理コーンとの等価性を検証する論理コーン比較手段とを備える。ここで、前記第1の論理コーン抽出手段は、シンボルシミュレーション手段を含んでいて良い。

【 0 0 3 4 】

より具体的には、本発明の第1の論理検証システムは、プログラム言語で記述された動作レベル記述をコンパイルして得たオブジェクトコードと、前記動作レベル記述から生成されたRTレベル記述と、前記動作レベル記述および前記RTレベル記述における等価性の比較の対象とする記述の組の情報とこの組毎の等価性の比較の対象となる信号の組の情報とを指定する対応情報と、前記動作レベル記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報とを記憶する記憶手段と、前記対応情報で指定される前記動作レベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる記述および信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行い、シンボルシミュレーションで得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして抽出する第1の論理コーン抽出手段と、前記対応情報で指定される前記RTレベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる信号毎の論理コーンを抽出する第2の論理コーン抽出手段と、前記対応情報で指定される前記動作レベル記述と前記RTレベル記述の比較の対象となる記述毎に、前記第1の論理コーン抽出手段で抽出された論理コーンと前記第2の論理コーン抽出手段で抽出された論理コ

ーンとを比較する論理コーン比較手段とを備えている。

【 0 0 3 5 】

また本発明の第 2 の論理検証システムは、プログラム言語で記述された動作レベル記述をコンパイルして得た CPU で実行可能なオブジェクトコードから論理コーンを抽出する第 1 の論理コーン抽出手段と、前記動作レベル記述が満たすべき性質であるプロパティを記憶する手段と、前記第 1 の論理コーン抽出手段で抽出された論理コーンをもとに、前記オブジェクトコードが前記プロパティを満たしているかどうかを検査するモデル検査手段とを備えている。

【 0 0 3 6 】

また本発明の論理コーン抽出装置は、プログラム記述をコンパイルして得たオブジェクトコードと、前記プログラム記述中の論理コーン抽出範囲および該論理コーン抽出範囲毎の抽出対象信号を指定する対応情報と、前記プログラム記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報とを入力する入力手段と、前記対応情報で指定される論理コーン抽出範囲毎に、前記対応情報で指定される論理コーン抽出範囲および抽出対象信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行うシンボルシミュレーション手段と、前記シンボルシミュレーション手段で得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして出力する出力手段とを含んで構成される。

【 0 0 3 7 】

また本発明の論理検証方法は、プログラム言語で記述された動作レベル記述をコンパイルして得た CPU で実行可能なオブジェクトコードから論理コーンを抽出する第 1 の論理コーン抽出ステップを含むことを基本とする。そして、RT レベル記述との論理検証に対して本発明を適用する場合、更に、RT レベル記述から論理コーンを抽出する第 2 の論理コーン抽出ステップと、前記第 1 の論理コーン抽出ステップで抽出された論理コーンと前記第 2 の論理コーン抽出ステップで抽出された論理コーンどうしの等価性を検証する論理コーン比較ステップとを含む

ように構成される。ここで、前記第 1 の論理コーン抽出ステップは、シンボルシミュレーションを含むものであって良い。

【 0 0 3 8 】

より具体的には、本発明の第 1 の論理検証方法は、プログラム言語で記述された動作レベル記述をコンパイルして得たオブジェクトコードと、前記動作レベル記述から生成された R T レベル記述と、前記動作レベル記述および前記 R T レベル記述における等価性の比較の対象とする記述の組の情報とこの組毎の等価性の比較の対象となる信号の組の情報とを指定する対応情報と、前記動作レベル記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報とを入力する入力ステップと、前記対応情報で指定される前記動作レベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる記述および信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行い、シンボルシミュレーションで得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして抽出する第 1 の論理コーン抽出ステップと、前記対応情報で指定される前記 R T レベル記述の比較の対象となる記述毎に、前記対応情報で指定される比較の対象となる信号毎の論理コーンを抽出する第 2 の論理コーン抽出ステップと、前記対応情報で指定される前記動作レベル記述と前記 R T レベル記述の比較の対象となる記述毎に、前記第 1 の論理コーン抽出ステップで抽出された論理コーンと前記第 2 の論理コーン抽出ステップで抽出された論理コーンとを比較する論理コーン比較ステップとを含んで構成される。

【 0 0 3 9 】

また本発明の第 2 の論理検証方法は、プログラム言語で記述された動作レベル記述をコンパイルして得た C P U で実行可能なオブジェクトコードから論理コーンを抽出する第 1 の論理コーン抽出ステップと、前記動作レベル記述が満たすべき性質であるプロパティを入力するステップと、前記第 1 の論理コーン抽出ステップで抽出された論理コーンをもとに、前記オブジェクトコードが前記プロパティ

を満たしているかどうかを検査する検査ステップとを含んで構成される。

【 0 0 4 0 】

また本発明の論理コーン抽出方法は、プログラム記述をコンパイルして得たオブジェクトコードと、前記プログラム記述中の論理コーン抽出範囲および該論理コーン抽出範囲毎の抽出対象信号を指定する対応情報と、前記プログラム記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報とを入力する入力ステップと、前記対応情報で指定される論理コーン抽出範囲毎に、前記対応情報で指定される論理コーン抽出範囲および抽出対象信号に対応する前記オブジェクトコードのコード部分および変数を前記コンパイル情報を参照して決定し、該決定した変数について、初期シンボル値を設定して、前記決定したコード部分の開始点から終了点までシンボルシミュレーションを行うシンボルシミュレーションのステップと、前記シンボルシミュレーションのステップで得られた前記変数のシンボルシミュレーション終了時点のシンボル値を前記変数の論理コーンとして出力する出力ステップとを含んで構成される。

【 0 0 4 1 】

【作用】

本発明の第 1 の論理検証システムおよび方法にあっては、CPU で直接実行されるオブジェクトコードから、論理等価性検証に必要な論理コーンを抽出することにより、プログラム記述言語を用いて記述された動作レベル記述からコンパイルされて得られたオブジェクトコードと RT レベル記述との等価性を検証することができる。これにより、動作レベル記述を CPU で実行したときの実行結果と、動作合成の結果得られる回路が動作したときの実行結果が完全に一致することを保証できるような論理検証が可能となる。

【 0 0 4 2 】

本発明の第 2 の論理検証システムおよび方法にあっては、CPU で直接実行されるオブジェクトコードから、論理等価性検証に必要な論理コーンを抽出することにより、CPU で実行される動作に対する形式的プロパティ検証が可能となる。

【 0 0 4 3 】

本発明の論理コーン抽出装置および方法にあっては、プログラム記述中の論理コ

ーン抽出範囲および該論理コーン抽出範囲毎の抽出対象信号を指定する対応情報と、前記プログラム記述と前記オブジェクトコードとのマッピング情報を含むコンパイル情報とに基づいて、前記プログラム記述をコンパイルして得たオブジェクトコードから、前記対応情報で指定した論理コーン抽出範囲毎に、前記対応情報で指定した抽出対象信号に対応する前記オブジェクトコードの変数毎の論理コーンを自動的に抽出することができる。

【 0 0 4 4 】

【発明の実施の形態】

次に、本発明の実施の形態について図面を参照して詳細に説明する。

【 0 0 4 5 】

図 1 を参照すると、本発明の第 1 の実施の形態は、情報を記憶する記憶装置 1 と、プログラム制御により動作するデータ処理装置 2 と、ディスプレイ装置や印刷装置等の出力装置 3 を含む。

【 0 0 4 6 】

データ処理装置 2 は、コンパイラ 2 1 と、オブジェクトコード論理コーン抽出手段 2 2 と、HDL 論理コーン抽出手段 2 3 と、論理コーン比較手段 2 4 とを含む。

【 0 0 4 7 】

記憶装置 1 は、コンパイルライブラリ 1 1 と、プログラム言語で表現された動作レベル記述 1 2 と、対応情報 1 3 と、HDL で表現された R T レベル記述 1 4 とをあらかじめ記憶している。

【 0 0 4 8 】

動作レベル記述 1 2 は一般に人手により記述される。その記述言語は、たとえば、C 言語、C++ 言語、j a v a 言語、S y s t e m C、S p e c C などである。コンパイルライブラリ 1 1 は、動作レベル記述 1 2 をオブジェクトコードにコンパイルして C P U で動作させるために必要になるライブラリであり、たとえば、S y s t e m C などの標準ライブラリであったり、設計者により作成されたり、コンパイラメーカーにより供給される。

【 0 0 4 9 】

R T レベル記述 1 4 は、たとえば、動作合成フェーズにより作成されるものであり、たとえば、動作合成システムに動作レベル記述 1 2 を与えることで得られる。あるいは、たとえば、動作レベル記述 1 2 から人手により変換されるものである。

【 0 0 5 0 】

対応情報 1 3 は、動作合成フェーズにおいて、動作レベル記述 1 2 を R T レベル記述 1 4 に変換する際に、動作合成システムもしくは、設計者によって与えられる。対応情報 1 3 は、等価性の比較の対象とする記述の組の情報と、この組毎の等価性の比較の対象となる信号の組の情報とを備えている。比較の対象とする記述の組の情報とは、動作レベル記述 1 2 の一部（もしくは全部）を特定する情報と、この動作レベル記述 1 2 の一部（もしくは全部）から合成される R T レベル記述 1 4 の一部（もしくは全部）を特定する情報である。たとえば、動作レベル記述 1 2 が一クロックサイクルの動作を順に記載している場合は、この一クロックサイクルの動作部分のそれぞれに対して、この記述から合成される R T レベル記述 1 4 の一部分が組の情報を備える。

【 0 0 5 1 】

この点を図 1 8 を参照して説明すると、動作レベル記述 1 2 に含まれる或る記述 A とその記述 A に対応する R T レベル記述 1 4 の記述 B とを比較の対象とする場合、対応情報 1 3 には、記述 A を特定する情報と記述 B を特定する情報との組（A : B）が設定される。また、記述 A に信号 a 1、a 2、記述 B に信号 b 1、b 2 がそれぞれあり、a 1 と b 1、a 2 と b 2、をそれぞれ比較の対象とする場合、対応情報 1 3 には、記述 A と記述 B との組の情報（A : B）に対応して、信号 a 1 と信号 b 1 との対応情報（a 1 : b 1）、信号 a 2 と信号 b 2 との対応情報（a 2 : b 2）が設定される。なお、検証の精度を確保するためには、等価性の比較の対象とする記述の組、その組の等価性の比較の対象となる信号の組、それぞれを網羅的に対応情報 1 3 に記述しておくことが望まれる。

【 0 0 5 2 】

コンパイラ 2 1 は、コンパイルライブラリ 1 1 と動作レベル記述 1 2 から C P U で実行可能なオブジェクトコード 1 5 およびコンパイル情報 1 6 を生成し、記憶

装置 1 に記憶する。コンパイル情報 1 6 は、たとえば、ELF(executable and linking format)といった、標準フォーマットで提供される。ここで、コンパイル情報 1 6 には、動作レベル記述 1 2 とオブジェクトコード 1 5 とのマッピング情報が含まれる。このマッピング情報により、動作レベル記述 1 2 のどの記述部分が、オブジェクトコード 1 5 のどのコード部分に対応しているか、動作レベル記述 1 2 のどの信号が、オブジェクトコード 1 5 のどの変数に対応しているか、が指示される。

【 0 0 5 3 】

この点を図 1 8 を参照して説明すると、動作レベル記述 1 2 の記述 A がオブジェクトコード 1 5 上のオブジェクトコード部分 C に対応する場合、コンパイル情報 1 6 には、記述 A がオブジェクトコード部分 C 1 に対応する旨の情報 (A : C) が含まれる。また、記述 A の信号 a 1 がオブジェクトコード 1 5 上では変数 c 1 で表現され、記述 A の信号 a 2 がオブジェクトコード 1 5 上では変数 c 2 で表現される場合、信号 a 1 と変数 c 1 の対応情報 (a 1 : c 1)、信号 a 2 と変数 c 2 の対応情報 (a 2 : c 2) がコンパイル情報 1 6 によって示される。

【 0 0 5 4 】

オブジェクトコード論理コーン抽出手段 2 2 は、記憶装置 1 に記憶されたオブジェクトコード 1 5 およびコンパイル情報 1 6 を読み込み、対応情報 1 3 で比較の対象とされた記述毎に、その記述内の比較の対象とされた信号に対応する変数毎の論理コーン (動作レベル論理コーン 1 7) を抽出する。抽出された動作レベル論理コーン 1 7 は記憶装置 1 に記憶する。

【 0 0 5 5 】

この点を図 1 8 を参照して説明すると、オブジェクトコード論理コーン抽出手段 2 2 は、対応情報 1 3 を参照して、記述 A が比較の対象となる記述の 1 つであり、この記述 A 中の信号 a 1、a 2 が比較の対象となる信号であることを認識し、コンパイル情報 1 6 を参照して、記述 A に対応するオブジェクトコード部分が C であり、信号 a 1、a 2 に対応する変数が c 1、c 2 であることを認識する。そして、オブジェクトコード部分 C に関して、後述するようにシンボルシミュレーションを行って、変数 c 1 の論理コーンと変数 c 2 の論理コーンとを抽出し、記

憶装置 1 に記憶する。

【 0 0 5 6 】

HDL 記述論理コーン抽出手段 2 3 は、記憶装置 1 に記憶された R T レベル記述 1 4 を読み込み、対応情報 1 3 で比較の対象とされた信号の論理コーン（R T レベル論理コーン 1 8）を抽出する。抽出された R T レベル論理コーン 1 8 は記憶装置 1 に記憶する。

【 0 0 5 7 】

この点を図 1 8 を参照して説明すると、HDL 記述論理コーン抽出手段 2 3 は、対応情報 1 3 を参照して、記述 B が比較の対象となる記述の 1 つであり、この記述中の信号 b 1、b 2 が比較の対象となる信号であることを認識し、R T レベル記述 1 4 から従来と同様にして信号 b 1 の論理コーンと信号 b 2 の論理コーンとを抽出し、記憶装置 1 に記憶する。

【 0 0 5 8 】

論理コーン比較手段 2 4 は、記憶装置 1 に記憶された対応情報 1 3 で示される比較の対象となる記述の組毎に、その組の記述に含まれる比較対象信号の組を選択し、記憶装置 1 からこの信号の論理コーン 1 7、1 8 をそれぞれ取り出して、双方の論理コーン 1 7、1 8 が論理的に等価であることを判定する。そして、判定結果を出力装置 3 から出力する。

【 0 0 5 9 】

この点を図 1 8 を参照して説明すると、論理コーン比較手段 2 4 は、対応情報 1 3 を参照して、動作レベル記述 1 2 の記述 A と R T レベル記述 1 4 の記述 B との組が比較の対象となる記述の組の 1 つであること、その組の記述 A、B に含まれる信号 a 1 と信号 b 1、信号 a 2 と信号 b 2 とがそれぞれ比較対象信号の組であることをそれぞれ認識する。また、コンパイル情報 1 6 を参照して、動作レベル記述 1 2 の記述 A がオブジェクトコード 1 5 の C に対応し、信号 a 1、a 2 が変数 c 1、c 2 に対応することを認識する。そして、記憶装置 1 から、変数 c 1 の論理コーンと信号 b 1 の論理コーンとを取り出し、双方の論理コーンが論理的に等価であるかどうかを判定する。また、記憶装置 1 から、変数 c 2 の論理コーンと信号 b 2 の論理コーンとを取り出し、双方の論理コーンが論理的に等価である

かどうかを判定する。

【0060】

次に、各図を参照して本実施の形態の動作について詳細に説明する。

【0061】

図2において、ステップA1では、動作レベル記述12よりコンパイルして得たオブジェクトコード15から、比較対象信号の論理コーン17を抽出する。このステップA1の詳細は後述する。ステップA2では、RTレベル記述14から、比較対象信号の論理コーン18を抽出する。論理コーンは、対応情報13に記述された「比較対象とする記述の組」毎に「比較対象とする信号の組」それぞれについて一組ずつ抽出され、これが比較単位となる。

【0062】

ステップA3では、対応する比較対象信号の論理等価性を比較する。かかるステップを比較単位の分だけ繰り返す。すべての比較対象信号について等価であることが示されたとき、オブジェクトコード15とRTレベル記述14は等価であると結論づけられる。

【0063】

図3を参照すると、ステップA1の詳細が示されている。

【0064】

ステップA11では、動作レベル記述12をコンパイラ21によってCPU上で実行可能なオブジェクトコード15にコンパイルする。なお、既にコンパイルされたオブジェクトコード15が存在する場合には、このステップA11は省略してよい。

【0065】

ステップA12では、コンパイルされたオブジェクトコード15上で、シンボルシミュレーションを行い、比較対象信号の論理コーン17を抽出する。

【0066】

図4を参照すると、ステップA12の詳細が示されている。

【0067】

ステップA121では、オブジェクトコード15上にシンボルシミュレーション

の開始点を見つける。シンボルシミュレーションの開始点は、対応情報 1 3 およびコンパイル情報 1 6 を参照して決定される。具体的には、対応情報 1 3 に記述されている「比較の対象とする記述の組」から、動作レベル記述 1 2 の一部分が指定される。コンパイル情報 1 6 からこの部分記述の開始点に対応するオブジェクトコード 1 5 上の開始点が指示され、これがシンボルシミュレーションの開始点となる。この点を図 1 8 を参照して説明すると、対応情報 1 3 から動作レベル記述 1 2 の記述 A が 1 つの比較対象の記述と認識され、コンパイル情報 1 6 から記述 A がオブジェクトコード部分 C に対応することが認識され、このオブジェクトコード部分 C の開始点をシンボルシミュレーションの開始点とする。なお、後述するシンボルシミュレーションの終了点は、このオブジェクトコード部分 C の終了点である。

【 0 0 6 8 】

ステップ A 1 2 2 では、比較対象信号に初期シンボル値を対応付ける。比較対象信号は、対応情報 1 3 およびコンパイル情報 1 6 によって指示される。シンボル値とは、変数の値を、整数などの実際の値ではなく、記号などのシンボルとして表現した値である。この点を図 1 8 を参照して説明すると、対応情報 1 3 から、動作レベル記述 1 2 の記述 A に信号 a 1、信号 a 2 が比較対象信号として含まれることを認識し、コンパイル情報 1 6 から信号 a 1、a 2 はオブジェクトコード部分 C では変数 c 1、c 2 に対応することを認識し、変数 c 1、c 2 をそれぞれ比較対象信号としてそれらに初期シンボル値を対応付ける。

【 0 0 6 9 】

ステップ A 1 2 3 では、オブジェクトコード 1 5 上の前記シンボルシミュレーション開始点から順にオブジェクトコード 1 5 をトレースしながらオブジェクトコード 1 5 の一命令ごとにその命令の定義に従ってシンボルシミュレーションを進める。

【 0 0 7 0 】

CPU で実行されるオブジェクトコード 1 5 は、メモリやレジスタといった記憶素子からの値の読み込み、これらの値をもとにした演算、演算結果の記憶素子への書き込み、条件分岐など実行順序の変更の組み合わせからなっている。演算に

使用される値にシンボル値が含まれるときはシンボル値演算をおこなう。シンボル値演算は、実際の演算を行うのではなく、演算結果を関数などによる表現式で表すものであり、この表現式もシンボル値という。このとき、演算に使用される値に即値が含まれるときは即値をシンボル値に変換してからシンボル値演算を行う。シンボル値演算の結果得られる演算結果を表すシンボル値は指定される記憶素子に記憶される。条件分岐の分岐条件がシンボル値である場合には、分岐条件シンボル値を記憶して、分岐があるものとなないものそれぞれトレースを続行する。分岐条件が即値である場合には、分岐演算の結果に従い、どちらか一方のトレースを続行する。オブジェクトコード 1 5 のトレースはオブジェクトコード 1 5 上のシンボルシミュレーション終了点に達するまで続ける。

【 0 0 7 1 】

シンボルシミュレーションの終了点に到達したとき、比較対象信号に関して或るシンボル値が得られる。このシンボル値は、具体的に言うと、オブジェクトコード 1 5 をシンボルシミュレーション開始点からシンボルシミュレーション終了点まで実行したときの比較対象信号のシンボルシミュレーション終了点時点の値を、比較対象信号のシンボルシミュレーション開始点時点で設定した初期シンボル値で表現したシンボル値である。このシンボル値は、初期シンボル値とシンボル値演算からなっており、初期シンボル値を回路の入力変数、シンボル値演算を論理演算に変換すると、組み合わせ論理式となる。こうして得られた論理式を、論理コーンとする（ステップ A 1 2 5）。

【 0 0 7 2 】

なお、分岐により、シンボルシミュレーション終了点に到達する経路が複数ある場合には、あらかじめ記憶された分岐条件シンボル値とシンボルシミュレーション終了点に達したときのシンボル値の論理式を作成し、マルチプレクサによってひとつのシンボル値にまとめる。そして、これを同様に論理式に変換し、これを、論理コーンとする。

【 0 0 7 3 】

対応情報 1 3 中に、動作レベル記述 1 2 に関する複数の比較対象の記述が含まれる場合、各記述毎に図 4 の処理が実行される。

【 0 0 7 4 】

【実施例】

次に、具体的な実施例を用いて本実施の形態の動作を説明する。

【 0 0 7 5 】

図 5 を参照すると C 言語で記述された動作レベル記述 1 2 の例が示されている。この記述では、関数 `addition()` の部分が動作合成によって回路に合成されることが指定されている（11行目）。関数 `addition()` は、関数が呼ばれるたびに変数 `in0` の値が順に累積加算されたものが変数 `out0` に格納されるように変数 `a, b` を用いて記述されている（12～18行目）。

【 0 0 7 6 】

図 6 を参照すると図 5 の動作レベル記述 1 2 を変換して得られる R T レベル記述 1 4 の例が示されている。動作レベル記述 1 2 中の関数 `addition()` が、`addition` という名前のモジュールとして実現されている（1～13行目）。変数 `in0` がモジュールの入力として実現されている（2行目）。変数 `out0` がモジュールの出力として実現されている（3行目）。関数の動作が 2 つのレジスタ `RG01` および `RG02` を用いて実現されている（10、11行目）。

【 0 0 7 7 】

図 8 を参照すると、対応情報 1 3 の一部が例示されている。対応情報 1 3 では動作レベル記述 1 2 中の変数 `in0`、`out0`、`a`、`b` と R T レベル記述 1 4 上の信号 `in0`、`out0`、`RG01`、`RG02` との対応関係が与えられている。また、対応情報 1 3 では、関数 `addition()` が動作合成により合成されたこと、および、関数 `addition()` がモジュール `addition` として実現されたことも別途与えられる。

【 0 0 7 8 】

いま、図 5 の動作レベル記述 1 2、図 6 の R T レベル記述 1 4、図 8 の対応情報 1 3 が与えられたとする。

【 0 0 7 9 】

動作レベル記述 1 2 をコンパイラ 2 1 により C P U で実行可能なオブジェクトコード 1 5 に変換する（ステップ A 1 1）。

【 0 0 8 0 】

図 7 を参照すると、オブジェクトコード 1 5 の一部が例示されている。movl, addl はそれぞれ CPU で実行される命令を表している。movl はメモリ空間の指定アドレス（第一引数）から指定レジスタ（第二引数）への値の読み込みをする命令である。addl は指定レジスタ上（第一引数）の値とメモリ空間の指定アドレス（第二引数）に格納された値との加算を計算し、指定レジスタ（第一引数）に計算結果を格納する命令である。in0, a, b, out0 はメモリ空間内のアドレス位置を表すラベルである。eax は CPU に備えられたレジスタを示している。

【 0 0 8 1 】

ラベル addition から始まる一連の命令によって、動作レベル記述 1 2 の関数 addition() の動作が実現されている。2 行目から 3 行目によって変数 a の値が変数 b の値と加算され加算結果が変数 b に格納される。4 行目から 5 行目によって変数 in0 の値が変数 a に格納される。6 行目から 7 行目によって、変数 b の値が変数 out0 にコピーされる。

【 0 0 8 2 】

図 9 を参照すると、コンパイル情報 1 6 の一部が例示されている。コンパイル情報 1 6 では、動作レベル記述 1 2 の各変数 in0, out0, a, b が、オブジェクトコード 1 5 中では、メモリ空間上で同名のラベルが付けられたアドレスに格納されることを示している。さらに、コンパイル情報 1 6 では、関数 addition() の処理が、図 7 に例示されたオブジェクトコードの処理に対応していることも、別途指定されている。

【 0 0 8 3 】

次に、図 7 のオブジェクトコード 1 5 から論理コーン 1 7 を抽出する（ステップ A 1 2）。論理コーン 1 7 の抽出では、まず、オブジェクトコード 1 5 からシンボルシミュレーション開始点を見つける。シンボルシミュレーション開始点は、対応情報 1 3 およびコンパイル情報 1 6 で指定される。図 7 に例示されたオブジェクトコードでは、ラベル addition の後の命令がシンボルシミュレーション開始点となる。

【 0 0 8 4 】

次に、シンボルシミュレーションの対象となる変数に初期シンボル値を設定する

。図 7 に例示されたオブジェクトコード 1 5 では、シンボルシミュレーションの対象となる変数は、対応情報 1 3 およびコンパイル情報 1 6 から判明する、変数 $a, b, in0, out0$ である。

【 0 0 8 5 】

図 1 0 を参照すると、初期シンボル値の設定が例示されている。変数 $a, b, in0, out0$ に対応するメモリ空間上のアドレスに対して、初期シンボル値 $a', b', in0', out0'$ がそれぞれ対応付けられている。本実施例では、変数名の後にダッシュ (') をつけて実行前の変数の値を表現するものとしている。

【 0 0 8 6 】

次に、オブジェクトコード 1 5 を順にトレースしながらシンボルシミュレーションを進める (ステップ A 1 2 3) 。

【 0 0 8 7 】

図 1 1 を参照すると、シンボルシミュレーションの様子が例示されている。開始点から 2 つの命令を実行した後は、変数 b の値を表すシンボル値が、変数 a, b の加算結果を表すシンボル値 “ $a' + b'$ ” になっている。また、シンボルシミュレーション終了点までシンボルシミュレーションが進んだときは、変数 a の値が、変数 $in0$ の値を表すシンボル値 $in0'$ に、変数 b および $out0$ の値が変数 a, b の加算結果を表すシンボル値 “ $a' + b'$ ” になっていることを示している。これらを論理コーンとする (ステップ A 1 2 5) 。図 1 2 を参照すると抽出された論理コーンが例示されている。

【 0 0 8 8 】

次に、RT レベル記述 1 4 から論理コーン 1 8 を抽出する (ステップ A 2) 。図 1 3 を参照すると RT レベル記述 1 4 から抽出された論理コーン 1 8 が例示されている。

【 0 0 8 9 】

最後に、対応する信号のペアに対して抽出された論理コーン 1 7、1 8 が等価であるかを調べる (ステップ A 3) 。図 8 および図 9 を参照すると、対応する信号のペアは図 1 4 に示される。すべてのペアにおいて論理コーン 1 7、1 8 が等価である場合は、オブジェクトコード 1 5 と RT レベル記述 1 4 とが等価であると

結論づける。

【 0 0 9 0 】

【発明の第 2 の実施の形態】

本発明の第 2 の実施の形態を、図面を用いて詳細に説明する。

【 0 0 9 1 】

本発明の第 2 の実施の形態では、R T レベル記述がH D Lで与えられる代わりに、プログラム言語で与えられる点異なる。

【 0 0 9 2 】

図 1 5 を参照すると、本発明の第 2 の実施の形態は、R T レベル記述 1 4 からのR T レベル論理コーン 1 8 の抽出を、図 1 に示されるH D L 論理コーン抽出手段 2 3 のかわりに、コンパイラ 2 1 A、オブジェクトコード論理コーン抽出手段 2 2 Aで行うことが異なる。

【 0 0 9 3 】

コンパイラ 2 1 A は、コンパイルライブラリ 1 1 A とプログラム言語で与えられたR T レベル記述 1 4 からC P U で実行可能なオブジェクトコード 1 A および、R T レベル記述 1 4 とオブジェクトコード 1 A との間の、変数とプログラムの格納位置の対応関係を記したコンパイル情報 1 B を生成し、記憶装置 1 に記憶する。

【 0 0 9 4 】

オブジェクトコード論理コーン抽出手段 2 2 A は、記憶装置 1 に記憶されたオブジェクトコード 1 A を読み込み、対応情報 1 3 とコンパイル情報 1 B を参照しながら、比較の対象とされた記述毎に、その記述に含まれる変数の論理コーン（動作レベル論理コーン 1 7 ）を抽出する。その動作は、オブジェクトコード論理コーン抽出手段 2 2 と基本的に同じである。

【 0 0 9 5 】

論理コーン比較手段 2 4 が動作レベル論理コーン 1 7 とR T レベル論理コーン 1 8 の等価性を比較することは、本発明の第 1 の実施の形態と同一である。

【 0 0 9 6 】

次に、本実施の形態の効果について説明する。

【 0 0 9 7 】

プログラム言語を用いて R T レベル記述をおこなうことで、プログラム言語で用意された豊富なライブラリを用いてシミュレーションを行うことができる。その場合、動作合成システムは H D L ではなくプログラム言語をもちいて R T レベル記述を出力するという要求も増えてきている。本実施の形態では、このように、R T レベル記述として H D L 記述ではなく、プログラム記述が与えられても、与えられた二つの記述の等価性を検証することができる。

【 0 0 9 8 】

【発明の第 3 の実施の形態】

本発明の第 3 の実施の形態を、図面を用いて詳細に説明する。

【 0 0 9 9 】

本発明の第 3 の実施の形態では、2 つの記述があたえられて、それらの等価性を検証するのではなく、与えられたプログラム記述が与えられたプロパティを満たしていることを証明する点が異なる。

【 0 1 0 0 】

図 1 6 を参照すると、本発明の第 3 の実施の形態は、モデル検査手段 2 5 を備え、形式的等価検証ではなく、形式的プロパティ検証を行う点で異なる。

【 0 1 0 1 】

コンパイラ 2 1、オブジェクトコード論理コーン抽出手段 2 2 を備え、動作レベル記述 1 2、コンパイルライブラリ 1 1、対応情報 1 3 から、動作レベル論理コーン 1 7 を抽出する点は、本発明の第一の実施の形態と同一である。

【 0 1 0 2 】

モデル検査手段 2 5 は、動作レベル論理コーン 1 7 と、プロパティ 1 C があたえられ、論理コーン 1 7 をもとに、回路がとりうる信号の値の組み合わせ（状態と呼ぶ）の集合を順に列挙する。回路がとりうるすべての状態の集合を列挙したところで、これらの集合が与えられたプロパティを満たしているかのチェックを行う。

【 0 1 0 3 】

図 1 7 を参照すると、本発明の第 3 の実施の形態の動作が示されている。

【0104】

ステップA1では、プログラム言語で記述された動作レベル記述12を、コンパイラ21でCPUが実行可能なオブジェクトコード15にコンパイルする。

【0105】

ステップA2では、コンパイルされたオブジェクトコード15から、論理コーン17を抽出する。

【0106】

ステップA3では、モデル検査手段25によって、論理コーン17をもとに、動作レベル記述12が与えられた性質を満足するかどうかを判定し、判定結果を出力装置3に出力する。モデル検査手段25には、たとえば前出の既存の技術を利用する。

【0107】

次に、本実施の形態の効果について説明する。本実施の形態では、RTレベル記述の代わりに、プログラム言語で記述された動作レベル記述12が満たすべきプロパティ1Cが与えられたときでも、動作レベル記述12がプロパティを満たすことを検証できる。

【0108】

以上本発明の実施の形態および実施例について説明したが、本発明は以上の実施の形態および実施例にのみ限定されるものでなく、その他各種の付加変更が可能である。また、本発明の論理検証システム及び論理コーン抽出装置は、その有する機能をハードウェア的に実現することは勿論、コンピュータと論理検証プログラム、論理コーン抽出プログラムとで実現することができる。論理検証プログラム、論理コーン抽出プログラムは、磁気ディスクや半導体メモリ等のコンピュータ可読記録媒体に記録されて提供され、コンピュータの立ち上げ時などにコンピュータに読み取られ、そのコンピュータの動作を制御することにより、そのコンピュータを前述した各実施の形態における論理検証システム及び論理コーン抽出装置として機能させる。

【0109】

【発明の効果】

以上説明したように本発明の第 1 の論理検証システムおよび方法によれば、プログラム記述言語を用いて記述された動作レベル記述からコンパイルされて得られたオブジェクトコードと R T レベル記述との等価性を検証することができる。これにより、動作レベル記述を C P U で実行したときの実行結果と、動作合成の結果得られる回路が動作したときの実行結果が完全に一致することを保証できるような論理検証が可能となる。

【 0 1 1 0 】

また本発明の第 2 の論理検証システムおよび方法によれば、プログラム記述言語を用いて記述された動作レベル記述からコンパイルされて得られたオブジェクトコードに対する形式的プロパティ検証が可能となる。

【 0 1 1 1 】

また本発明の論理コーン抽出装置および方法によれば、対応情報とコンパイル情報とに基づいて、プログラム記述をコンパイルして得たオブジェクトコードから、前記対応情報で指定した論理コーン抽出範囲毎に、前記対応情報で指定した抽出対象信号に対応する前記オブジェクトコードの変数毎の論理コーンを自動的に抽出することができる。

【図面の簡単な説明】

【図 1】

本発明の第 1 の実施の形態の構成を示すブロック図である。

【図 2】

第 1 の実施の形態の動作を示す流れ図である。

【図 3】

第 1 の実施の形態の動作を示す流れ図である。

【図 4】

第 1 の実施の形態の動作を示す流れ図である。

【図 5】

第 1 の実施の形態の動作の具体例を示す図である。

【図 6】

第 1 の実施の形態の動作の具体例を示す図である。

【図 7】

第 1 の実施の形態の動作の具体例を示す図である。

【図 8】

第 1 の実施の形態の動作の具体例を示す図である。

【図 9】

第 1 の実施の形態の動作の具体例を示す図である。

【図 1 0】

第 1 の実施の形態の動作の具体例を示す図である。

【図 1 1】

第 1 の実施の形態の動作の具体例を示す図である。

【図 1 2】

第 1 の実施の形態の動作の具体例を示す図である。

【図 1 3】

第 1 の実施の形態の動作の具体例を示す図である。

【図 1 4】

第 1 の実施の形態の動作の具体例を示す図である。

【図 1 5】

本発明の第 2 の実施の形態の構成を示すブロック図である。

【図 1 6】

本発明の第 3 の実施の形態の構成を示すブロック図である。

【図 1 7】

第 3 の実施の形態の動作を示す流れ図である。

【図 1 8】

本発明の第 1 の実施の形態の動作説明図である。

【図 1 9】

トップダウン設計の流れの概略を示す図である。

【図 2 0】

従来の形式的等価性検証装置のブロック図である。

【図 2 1】

従来の形式的プロパティ検証装置のブロック図である。

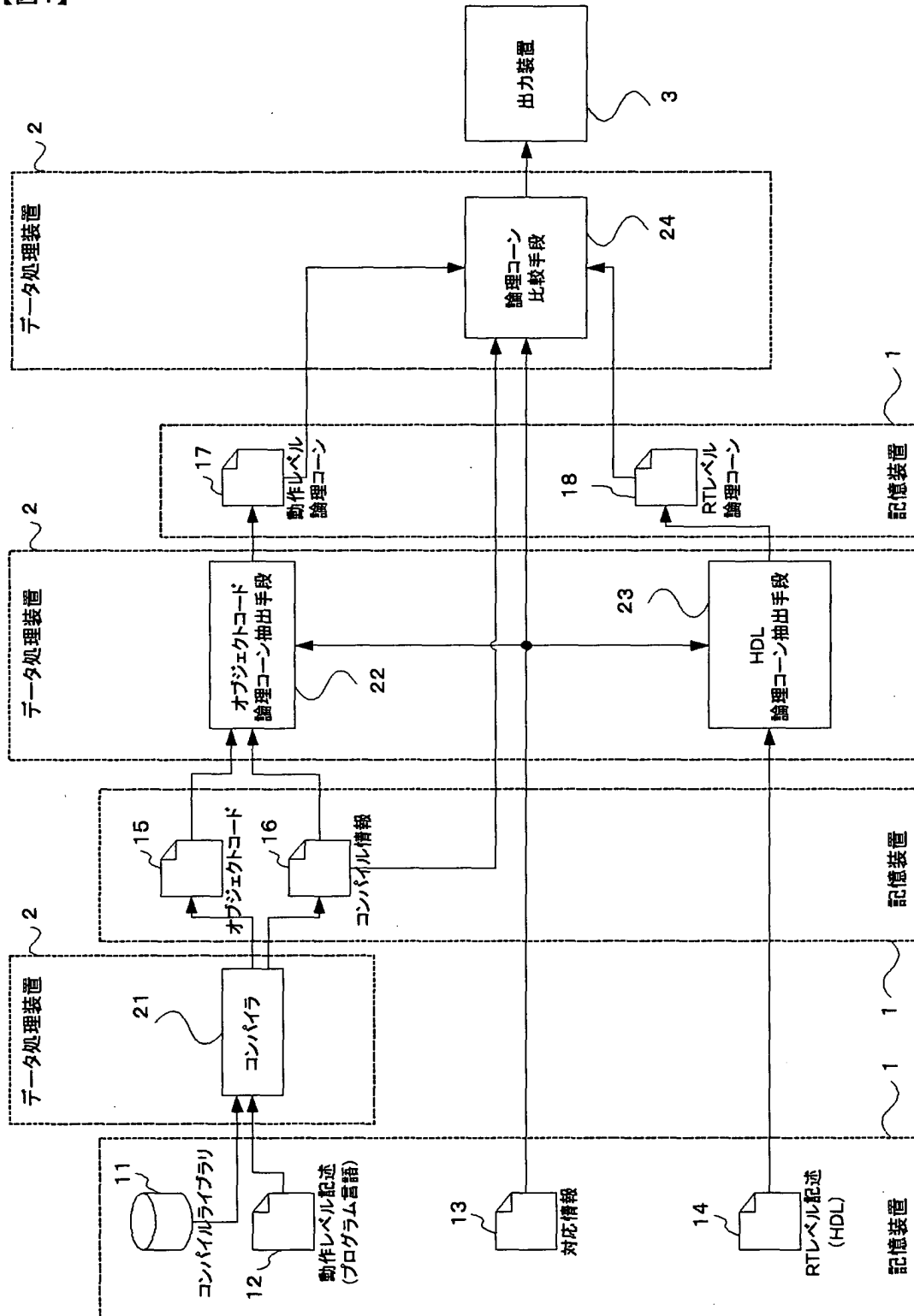
【符号の説明】

- 1 …記憶装置
- 2 …データ処理装置
- 3 …出力装置
- 1 1、1 1 A …コンパイルライブラリ
- 1 2 …動作レベル記述
- 1 3 …対応情報
- 1 4 …R T レベル記述
- 1 5、1 A …オブジェクトコード
- 1 6、1 B …コンパイル情報
- 1 7 …動作レベル論理コーン
- 1 8 …R T レベル論理コーン
- 1 C …プロパティ
- 1 D …ゲートレベル記述
- 1 E …ゲートレベル論理コーン
- 2 1、2 1 A …コンパイラ
- 2 2、2 2 A …オブジェクトコード論理コーン抽出手段
- 2 3 …H D L 論理コーン抽出手段
- 2 4 …論理コーン比較手段

【書類名】 図面

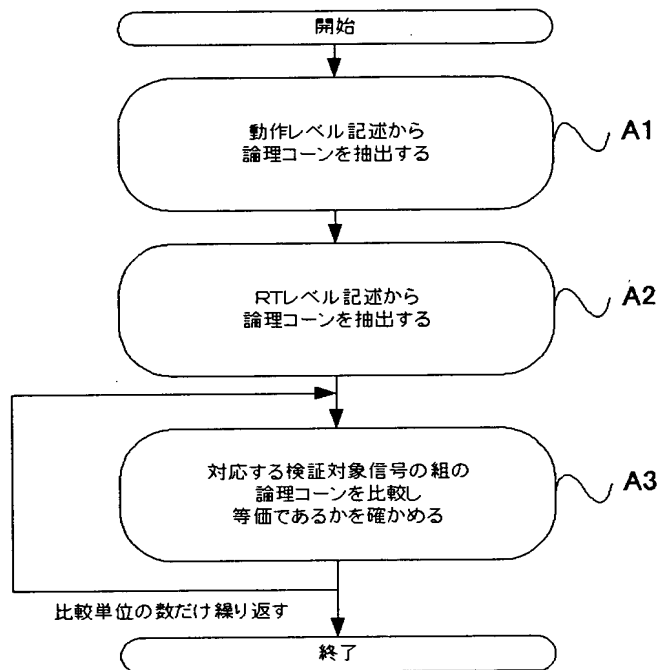
【図 1】

【図 1】



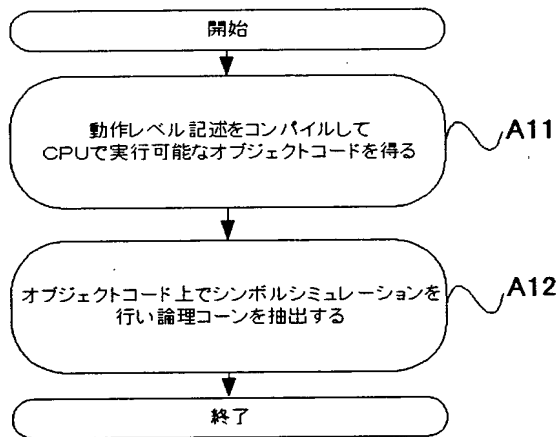
【図 2】

【図2】



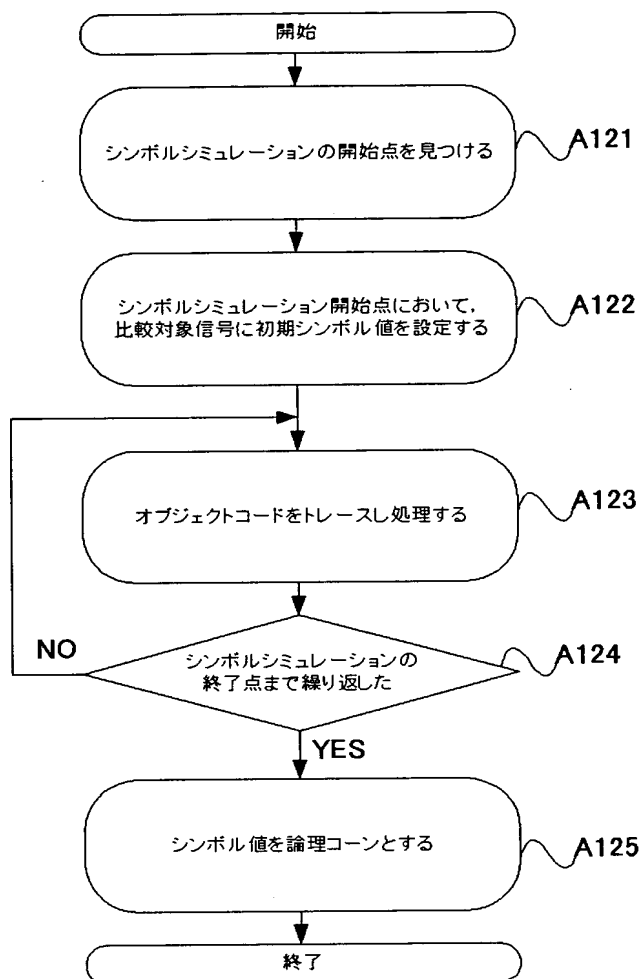
【図 3】

【図3】



【図 4】

【図 4】



【図5】

【図5】

12
↙

```

int a, b, c ; .....(1)
int in0, out0 ; .....(2)
#ifdef C .....(3)
main() { .....(4)
    while(1) { .....(5)
        scanf("%d", &in0) ; .....(6)
        addition(); .....(7)
        printf("%d\n", out0) ; .....(8)
    } .....(9)
} .....(10)
#endif .....(10)

/* to behavioral Synthesis */ .....(11)
void additon() .....(12)
{ .....(13)
    b = a + b ; .....(14)
    a = in0 ; .....(15)
    out0 = b ; .....(16)
    return ; .....(17)
} .....(18)

```

【図6】

【図6】

14
↙

```

module additon(in0, out0, CLOCK) ; .....(1)
input [31:0] in0 ; .....(2)
output [31:0] out0 ; .....(3)
input CLOCK .....(4)
reg[31:0] RG01; .....(5)
reg[31:0] RG02; .....(6)

assign out0 = RG02 ; .....(7)

always @ ( posedge CLOCK ) .....(8)
begin .....(9)
    RG01 <= in0 ; .....(10)
    RG02 <= RG01 + RG02 ; .....(11)
end .....(12)
endmodule .....(13)

```


【図 7】

【図7】

15

↙

```

addition:          .....(1)
    movl a,%eax     .....(2)
    addl %eax,b     .....(3)
    movl in0,%eax   .....(4)
    movl %eax,a     .....(5)
    movl b,%eax     .....(6)
    movl %eax,out0  .....(7)
    
```

【図 8】

【図 8】 対応情報の例（一部）

13

↙

C 言語記述上の信号	HDL 記述上の信号
in0	in0
out0	out0
a	RG01
b	RG02

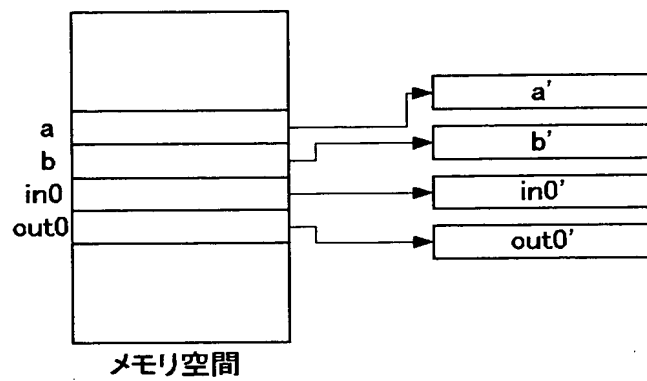
【図 9】

【図 9】 コンパイル情報の例（一部）

C 言語記述上の信号	オブジェクトコード上の記憶領域
in0	in0
out0	out0
a	a
b	b

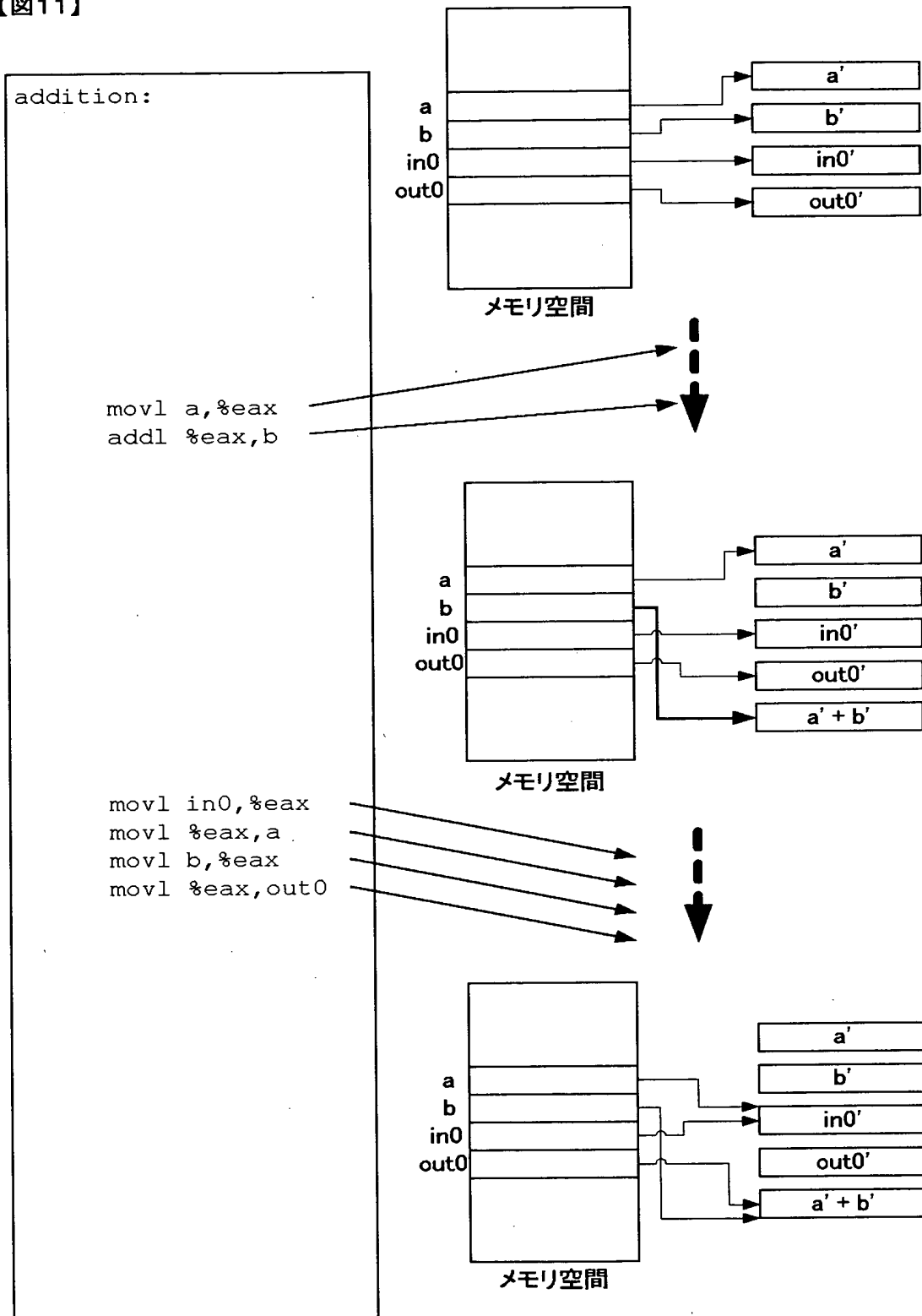
【図 1 0】

【図10】



【図 11】

【図11】



【図 1 2】

【図 1 2】 プログラム記述の論理コーンの例

変数	論理コーン
a	$\text{in0}'$
b	$a' + b'$
in0	$\text{in0}'$
out0	$a' + b'$

【図 1 3】

【図 1 3】 HDL 記述の論理コーンの例

変数	論理コーン
RG01	$\text{in0}'$
RG02	$\text{RG01}' + \text{RG02}'$
in0	$\text{in0}'$
out0	$\text{RG01}' + \text{RG02}'$

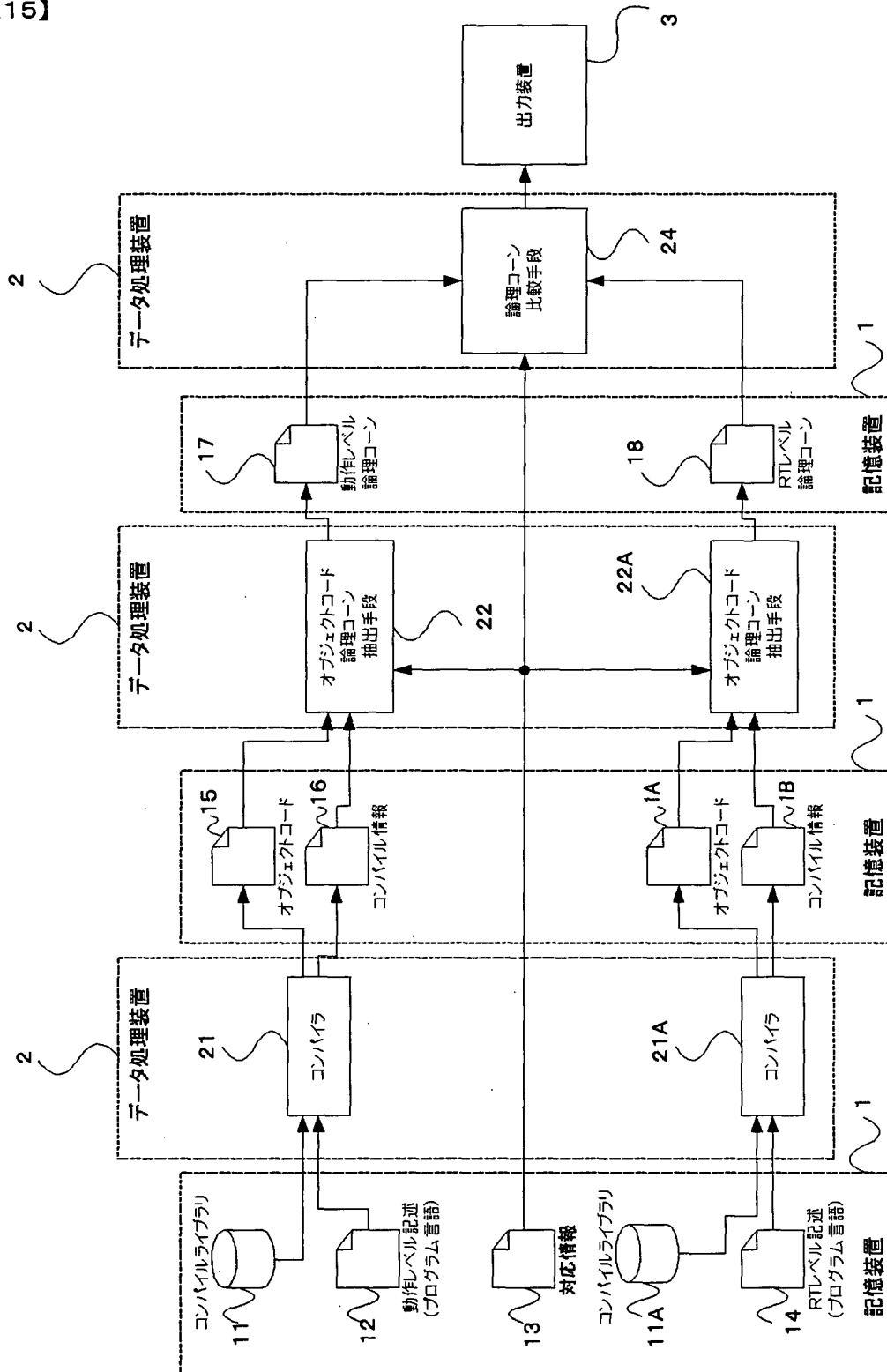
【図 1 4】

【図 1 4】 オブジェクトコード上の変数とHDL上の信号の対応

オブジェクトコード上の変数	HDL上の信号
a	RG01
b	RG02
in0	in0
out0	out0

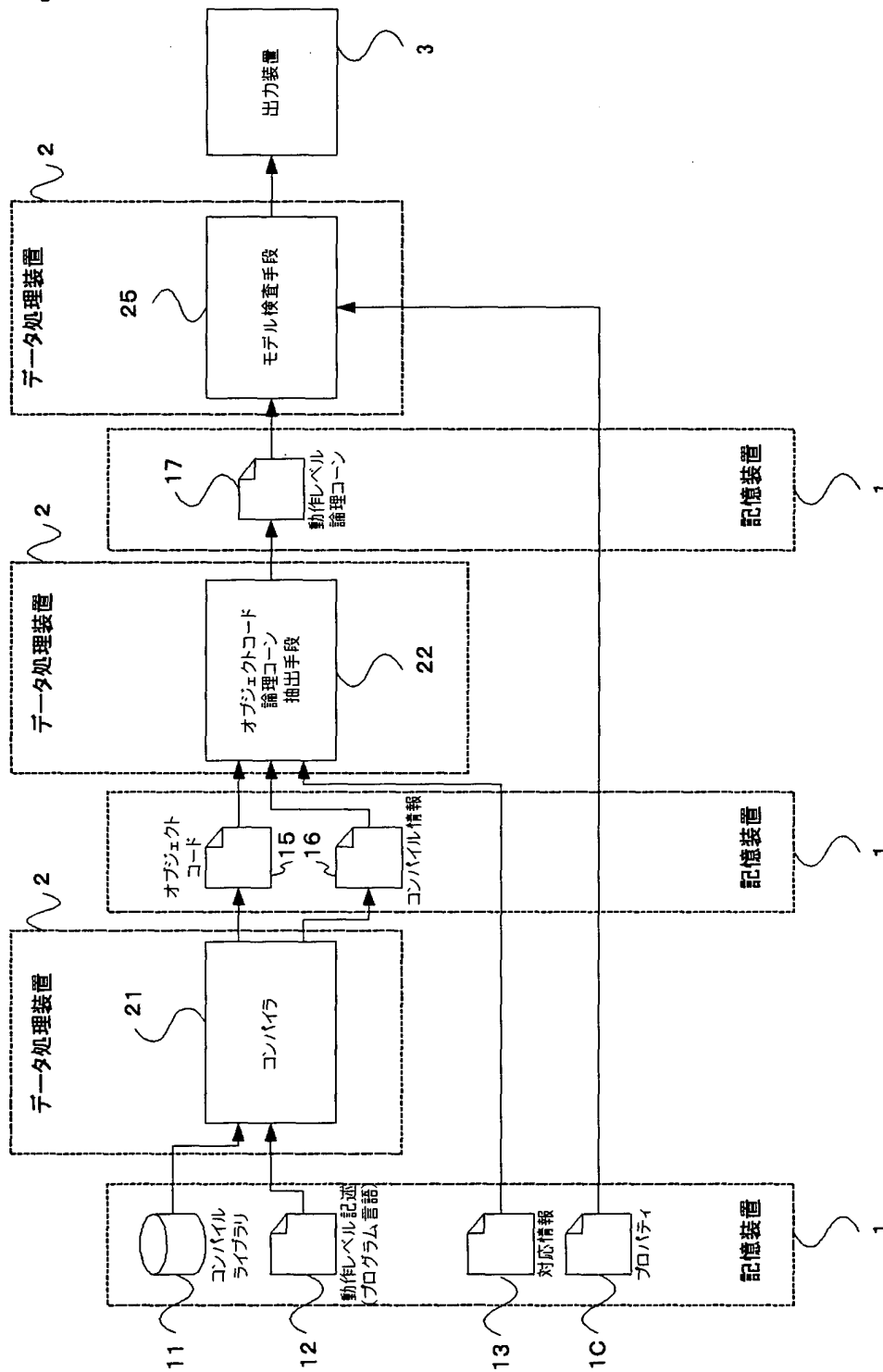
【図15】

【図15】



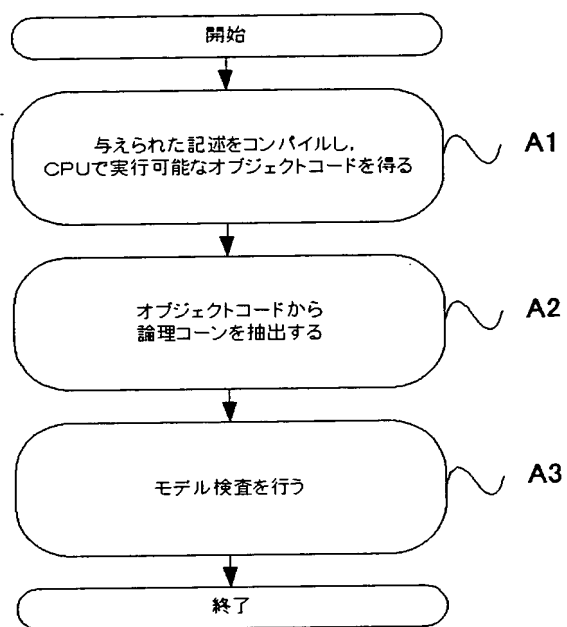
【図16】

【図16】



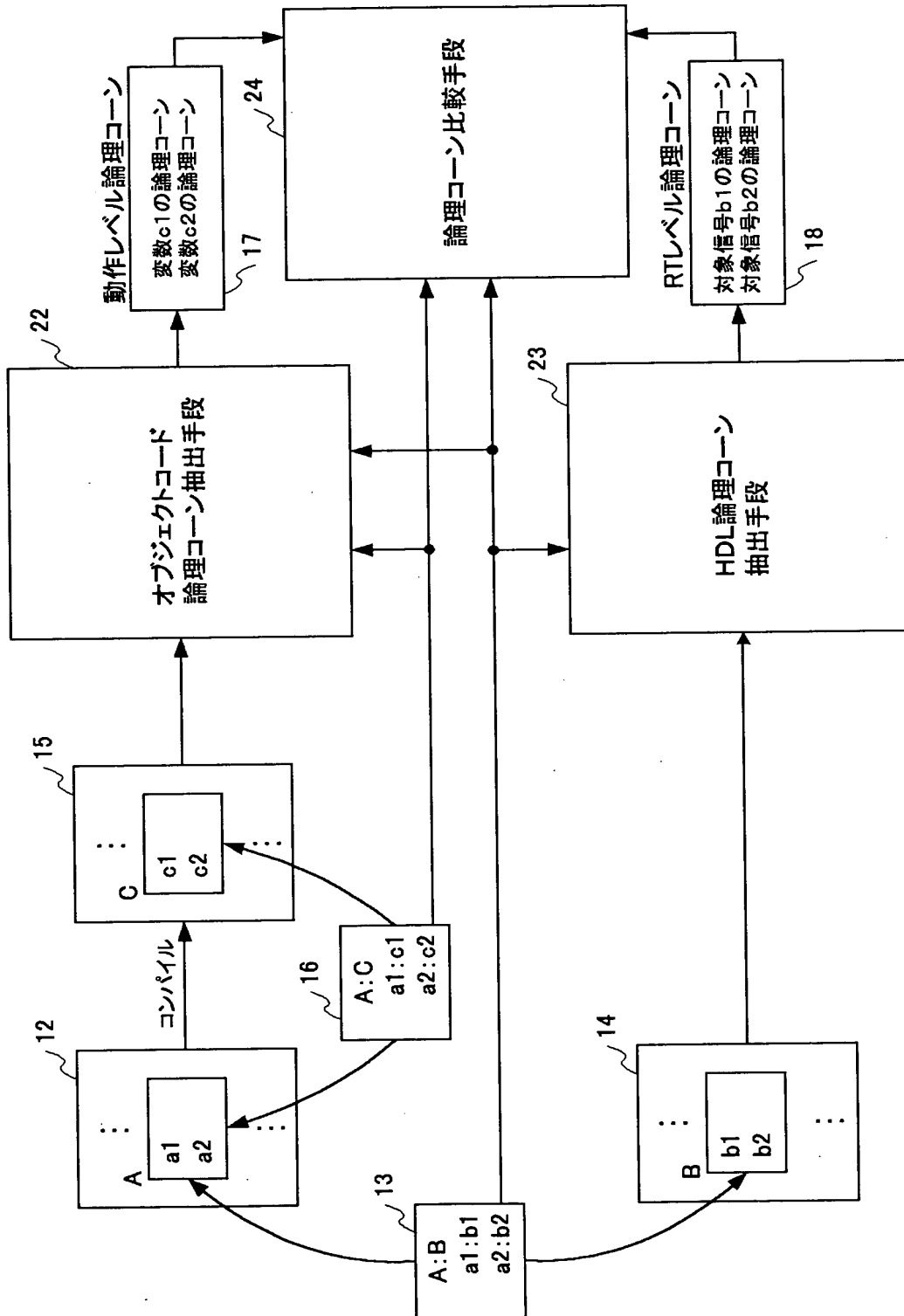
【図 1 7】

【図17】



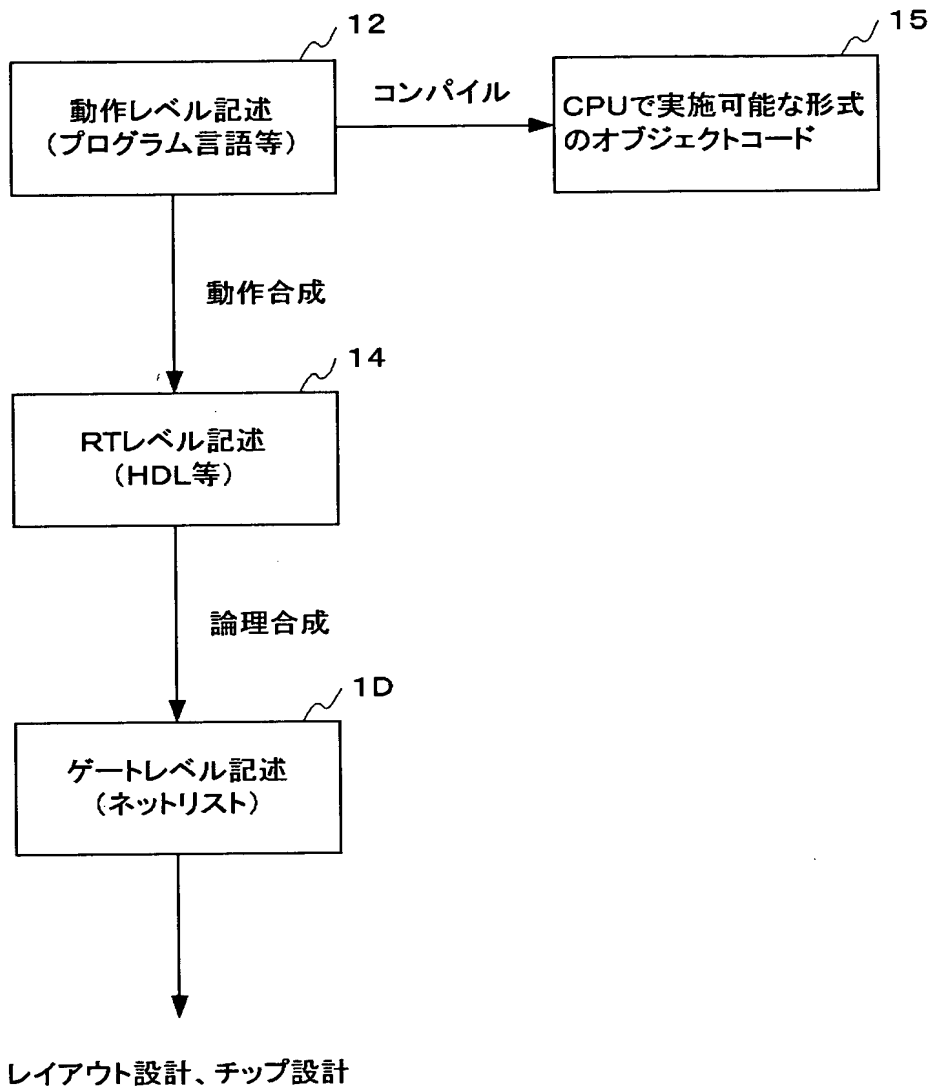
【図18】

【図18】



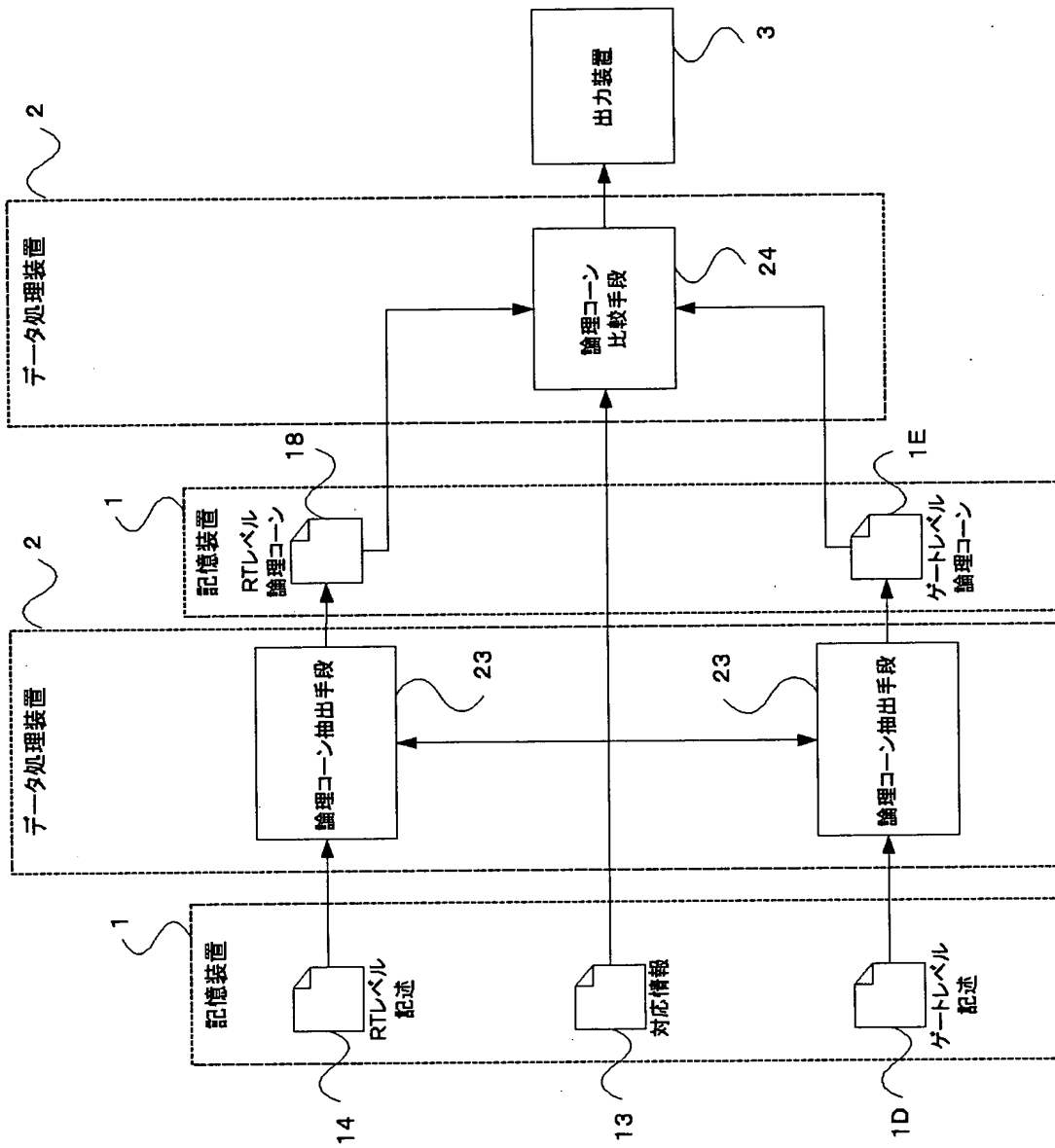
【図 1 9】

【図19】



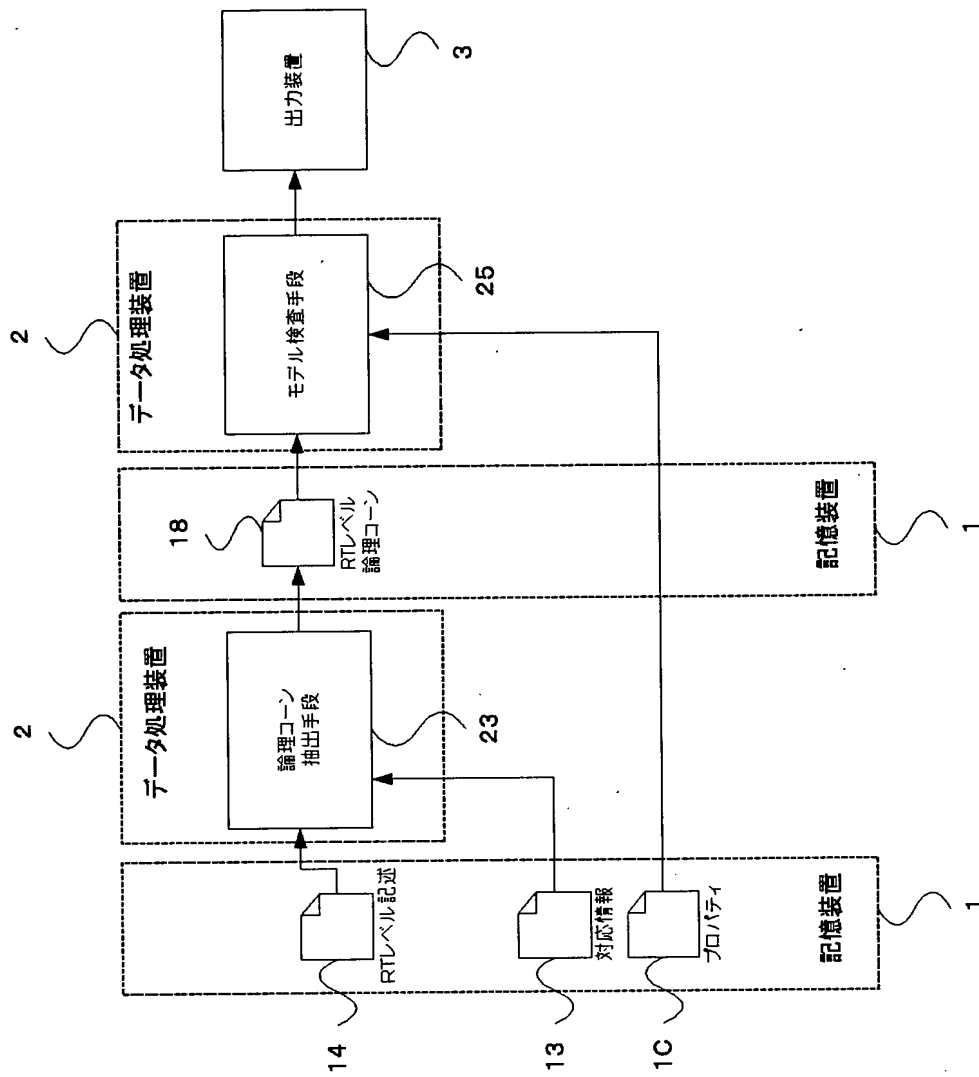
【図20】

【図20】



【図 21】

【図21】



【書類名】 要約書

【要約】

【課題】 動作レベル記述をコンパイルしたオブジェクトコードと動作レベル記述から導出した R T レベル記述との等価性を検証する。

【解決手段】 対応情報 1 3 は、プログラム言語で記述された動作レベル記述 1 2 及びそれから動作合成した R T レベル記述 1 4 における等価性の比較の対象とする記述の組の情報とこの組毎の等価性の比較の対象となる信号の組の情報とを指定する。コンパイル情報 1 6 は、動作レベル記述 1 2 とオブジェクトコード 1 5 とのマッピング情報を含む。論理コーン抽出手段 2 3 は、対応情報 1 3 及びコンパイル情報 1 6 を参照して、オブジェクトコード 1 5 から、シンボルシミュレーションを行って、論理コーン 1 7 を抽出する。論理コーン抽出手段 2 3 は、R T レベル記述 1 4 から論理コーン 1 8 を抽出する。論理コーン比較手段 2 4 は、論理コーン 1 7 と論理コーン 1 8 どうしの等価性を検証する。

【選択図】 図 1

出 願 人 履 歴 情 報

識別番号 [0 0 0 0 0 4 2 3 7]

1. 変更年月日	1 9 9 0 年 8 月 2 9 日
[変更理由]	新規登録
住 所	東京都港区芝五丁目 7 番 1 号
氏 名	日本電気株式会社